

Vorwort	XI
1 Gerade genug Microservices	1
Was sind Microservices?	1
Unabhängige Deploybarkeit	2
Modellierung rund um eine Businessdomäne.	2
Die eigenen Daten besitzen.	5
Welche Vorteile können Microservices haben?	6
Welche Probleme werden entstehen?	7
Benutzeroberflächen	8
Technologie	8
Größe	9
Und Ownership	11
Der Monolith.	12
Der Ein-Prozess-Monolith	12
Der verteilte Monolith	15
Black-Box-Systeme von Fremdherstellern	15
Herausforderungen von Monolithen	15
Vorteile von Monolithen.	16
Zu Kopplung und Kohäsion	17
Kohäsion.	18
Kopplung	18
Gerade genug Domain-Driven Design.	28
Aggregat	29
Kontextgrenzen.	31
Aggregate und Kontextgrenzen auf Microservices abbilden.	31
Weitere Quellen	32
Zusammenfassung.	32

2 Eine Migration planen	33
Das Ziel verstehen	33
Drei zentrale Fragen	35
Warum wollen Sie Microservices einsetzen?	35
Teamautonomie verbessern	35
Time-to-Market verringern	37
Kostengünstig auf Last reagieren	37
Robustheit verbessern	38
Die Anzahl der Entwickler erhöhen	40
Neue Technologien einsetzen	41
Wann können Microservices eine schlechte Idee sein?	43
Unklare Domäne	43
Start-ups	44
Beim Kunden installierte und verwaltete Software	45
Keinen guten Grund haben!	46
Abwägungen	46
Die Menschen mitnehmen	48
Organisationen verändern	48
Gefühl für die Dringlichkeit vermitteln	49
Führungscoalition schaffen	49
Vision und Strategie entwickeln	50
Veränderungsvision kommunizieren	51
Mitarbeitern umfangreiche Unterstützung ermöglichen	52
Kurzfristige Erfolge erzielen	53
Nutzen konsolidieren und weitere Veränderungen anstoßen	53
Neue Ansätze in der Unternehmenskultur verankern	54
Die Wichtigkeit der inkrementellen Migration	54
Nur die Produktivumgebung zählt	55
Veränderungskosten	55
Reversible und irreversible Entscheidungen	56
Bessere Orte zum Experimentieren	57
Wo fangen wir also an?	58
Domain-Driven Design	58
Wie weit müssen Sie gehen?	59
Event Storming	60
Ein Domänenmodell zum Priorisieren einsetzen	60
Ein kombiniertes Modell	62
Teams reorganisieren	64
Sich verändernde Strukturen	64
Es gibt nicht die eine Lösung für alle	65
Eine Änderung vornehmen	67
Veränderte Fähigkeiten	69

Woher wissen Sie, ob die Transformation funktioniert?	72
Regelmäßige Checkpoints	72
Quantitative Messgrößen	73
Qualitative Messwerte	73
Vermeiden Sie den Sunk-Cost-Effekt	74
Seien Sie offen für neue Ansätze	74
Zusammenfassung	75
3 Den Monolithen aufteilen	77
Ändern wir den Monolithen, oder lassen wir es bleiben?	77
Ausschneiden, einfügen oder reimplementieren?	78
Den Monolithen refaktorisieren	79
Migrations-Patterns	80
Pattern: Strangler Fig Application	81
Wie es funktioniert	81
Wo wir es einsetzen	83
Beispiel: HTTP Reverse Proxy	85
Daten?	88
Proxy-Optionen	88
Protokolle wechseln	91
Beispiel: FTP	94
Beispiel: Message Interception	95
Andere Protokolle	98
Andere Beispiele für das Strangler Fig Pattern	98
Verhaltensänderung während der Migration	99
Pattern: UI Composition	100
Beispiel: Page Composition	100
Beispiel: Widget Composition	101
Beispiel: Micro Frontends	104
Wo wir es einsetzen	105
Pattern: Branch by Abstraction	106
Wie es funktioniert	106
Als Fallback-Mechanismus	113
Wo wir es einsetzen	114
Pattern: Parallel Run	114
Beispiel: Preisbildung von Kreditderivaten	115
Beispiel: Homegate-Angebote	116
Verifikationstechniken	117
Spione einsetzen	117
Scientist von GitHub	119
Dark Launching und Canary Releasing	119
Wo wir es einsetzen	119

Pattern: Decorating Collaborator	120
Beispiel: Loyalty-Programm	120
Wo wir es einsetzen	121
Pattern: Change Data Capture	122
Beispiel: Loyalty-Karten ausgeben	122
Change Data Capture implementieren	123
Wo wir es einsetzen	126
Zusammenfassung	126
4 Die Datenbank aufteilen	127
Pattern: Shared Database	127
Hilfreiche Patterns	128
Wo wir es einsetzen	129
Aber es geht nicht!	129
Pattern: Database View	131
Die Datenbank als öffentlicher Vertrag	131
Präsentations-Views	132
Grenzen	133
Ownership	134
Wo wir es einsetzen	134
Pattern: Database Wrapping Service	134
Wo wir es einsetzen	136
Pattern: Database-as-a-Service Interface	137
Eine Mapping Engine implementieren	139
Vergleich mit Views	139
Wo wir es einsetzen	139
Ownership transferieren	140
Pattern: Aggregate Exposing Monolith	140
Pattern: Change Data Ownership	143
Datensynchronisation	144
Pattern: Synchronize Data in Application	146
Schritt 1: Daten Bulk-synchronisieren	146
Schritt 2: Synchrones Schreiben, aus dem alten Schema lesen	147
Schritt 3: Synchrones Schreiben, aus dem neuen Schema lesen	148
Wo dieses Pattern genutzt werden kann	148
Wo wir es verwenden	149
Pattern: Tracer Write	150
Datensynchronisierung	152
Beispiel: Bestellungen bei Square	154
Wo wir es verwenden	158
Die Datenbank aufteilen	158
Physische versus logische Datenbanktrennung	158

Zuerst die Datenbank oder zuerst den Code aufteilen?	160
Zuerst die Datenbank aufteilen.	161
Zuerst den Code aufteilen.	165
Datenbank und Code gleichzeitig aufteilen	169
Was sollte ich also als Erstes aufteilen?	170
Beispiele zur Schemaaufteilung	170
Pattern: Split Table	171
Wo wir es verwenden	173
Pattern: Move Foreign-Key Relationship to Code.	173
Den Join ersetzen	174
Datenkonsistenz	176
Wo wir es verwenden	178
Beispiel: Gemeinsam genutzte statische Daten.	178
Transaktionen	187
ACID-Transaktionen	187
Weiterhin ACID, aber ohne Atomarität?	188
Zwei-Phasen-Commit	190
Verteilte Transaktionen: Sagen Sie einfach Nein!	193
Sagas	193
Fehlersituationen in Sagas	195
Saga implementieren	199
Saga versus verteilte Transaktionen	205
Zusammenfassung	206
5 Wachsende Probleme	207
Mehr Services – mehr Schmerzen	207
Ownership im großen Maßstab.	209
Wie kann sich dieses Problem zeigen?	209
Wann kann sich das Problem zeigen?	210
Mögliche Lösungen.	210
Disruptive Änderungen	210
Wie kann sich dieses Problem zeigen?	211
Wann kann sich das Problem zeigen?	211
Mögliche Lösungen.	212
Reporting	215
Wann kann sich dieses Problem zeigen?	216
Mögliche Lösungen.	216
Monitoring und Troubleshooting	217
Wann kann sich dieses Problem zeigen?	218
Wie kann sich das Problem zeigen?	218
Mögliche Lösungen.	218

Lokale Entwicklung	222
Wie kann sich dieses Problem zeigen?	223
Wann kann sich das Problem zeigen?	223
Mögliche Lösungen	223
Zu viele Dinge laufen lassen	224
Wie kann sich dieses Problem zeigen?	224
Wann kann sich das Problem zeigen?	225
Mögliche Lösungen	225
End-to-End-Tests	226
Wie kann sich dieses Problem zeigen?	227
Wann kann sich das Problem zeigen?	227
Mögliche Lösungen	227
Globale versus lokale Optimierung	229
Wie kann sich dieses Problem zeigen?	229
Wann kann sich das Problem zeigen?	230
Mögliche Lösungen	230
Robustheit und Resilienz	232
Wie kann sich dieses Problem zeigen?	232
Wann kann sich das Problem zeigen?	232
Mögliche Lösungen	233
Verwaiste Services	233
Wie kann sich dieses Problem zeigen?	234
Wann kann sich das Problem zeigen?	234
Mögliche Lösungen	234
Zusammenfassung	236
Abschließende Worte	237
A Literatur	239
B Pattern-Index	241
Index	243